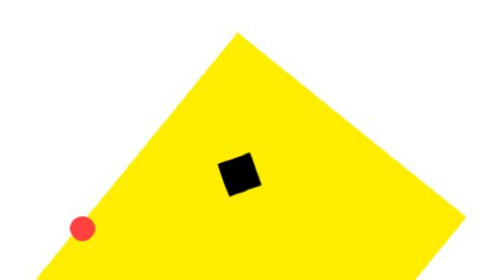
Многопоточность на фронте. Абсурд или прекрасное архитектурное решение?

Игорь Костяков





Вступление

ThreadA
Process 3
ThreadB

ThreadA
ThreadB
Process 1
ThreadC

ThreadA
Process 2

operating system

IBS Dunice



О себе



Игорь Костяков

Технический директор
IBS Dunice

- Разрабатываю на JS с 15 лет
- Ответственен за технологии, стандарты и технический уровень сотрудников в компании
- Популяризирую ценность знаний в команде
- Считаю, что все разумно в меру, и выбираю инструменты, исходя из задач





300+

SPECIALISTS AVAILABLE IN-HOUSE

2012

WAS FOUNDED

400+

PROJECTS



Dunice

WEB & MOBILE DEVELOPMENT













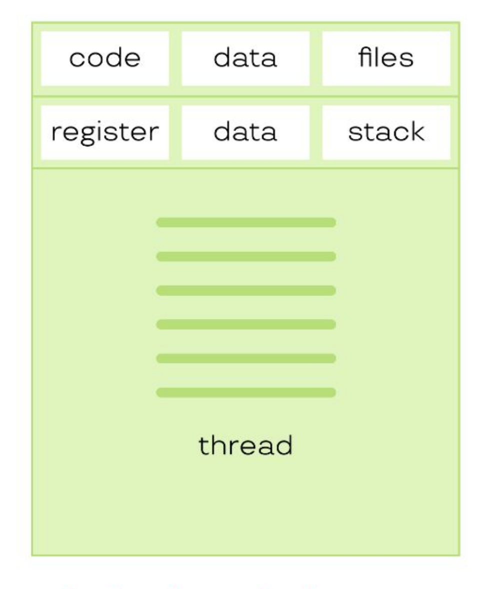




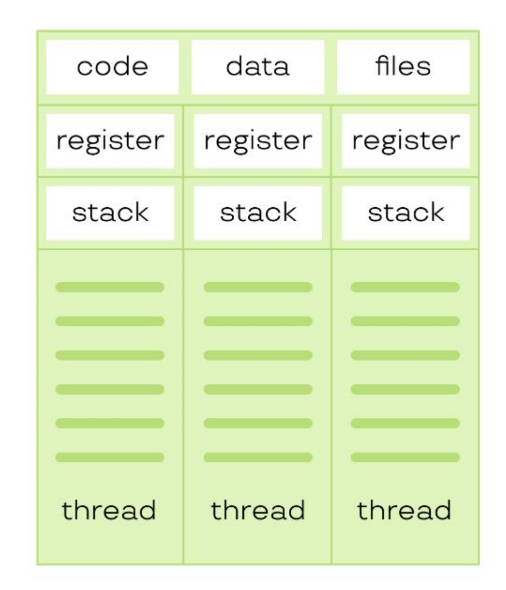




Многопоточность в браузере



single-threaded process



multithreaded process

Обзор dedicated (web) worker

main.js
Main Page / App

Non Bloking Code {

}

worker.js
Background Task

new Worker ( ) →
start →
← progress
← progress
← progress
← done !
worker . terminate ( ) →





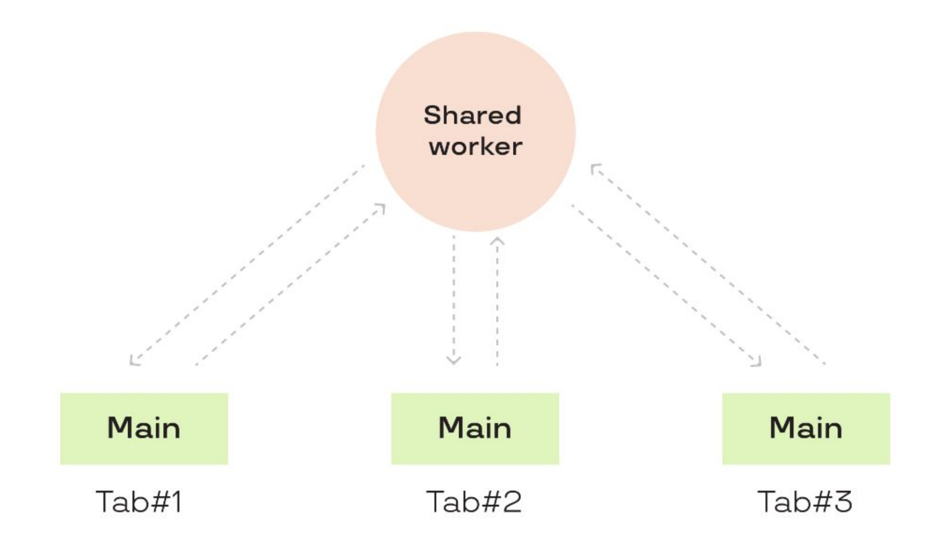
Обзор dedicated worker

Dedicated worker
Dedicated worker
Main
Tab#1

Dedicated worker
Main
Tab#2

Dedicated worker
Main
Tab#3

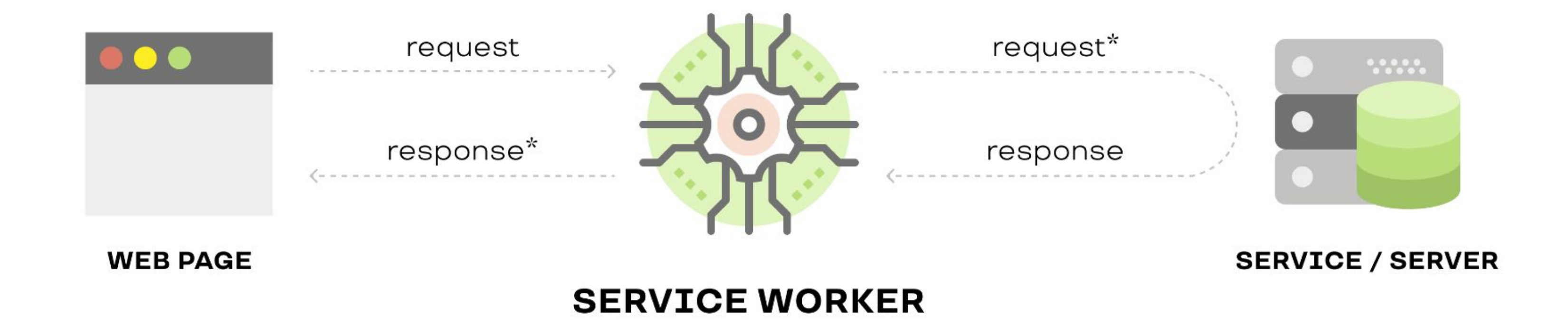
Обзор shared worker







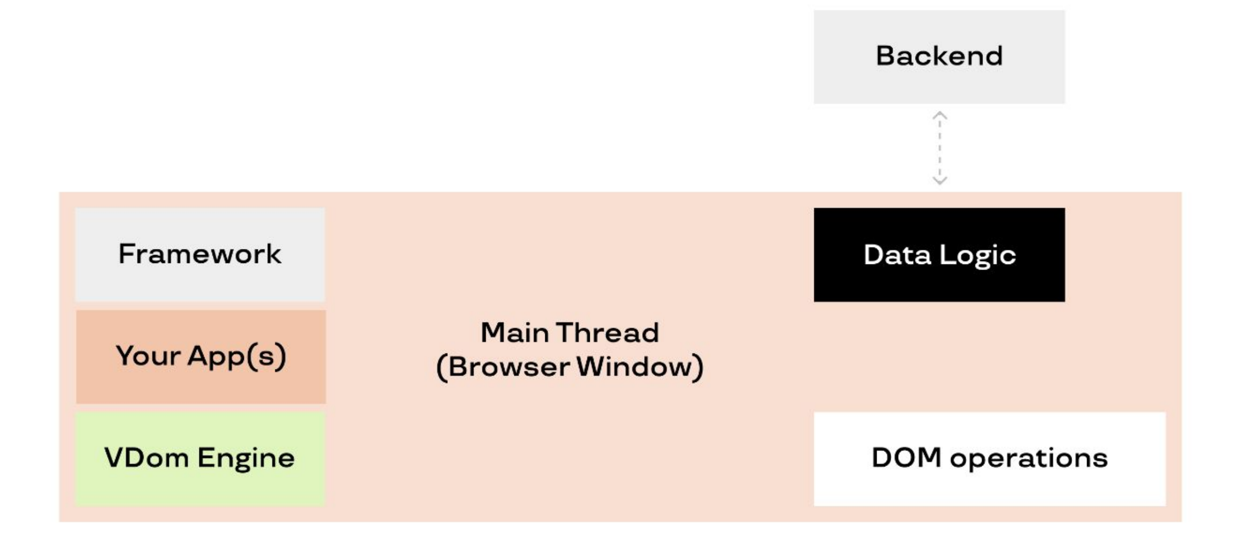
Обзор service worker







Чуть ближе к реальности



github.com/marketplace

Search or jump to... Pull requests Issues Marketplace Explore

xtend GitHub

d tools to improve your workflow

Explore free apps

Search for apps and actions

Apps

CircleCI
By circleci
Automatically build, test, and deploy your project in minutes
Recommended

Rewind Backups for GitHub (Formerly BackHub)
By backhub
Daily, automatic backups of your repos and metadata. Restore your backups with

html body.logged-in.env-production.page-responsive.color-bg-subtle

```
--sub-nav-mktg-shadow: 0 0 0 1px
  hsl(210deg 26% 28% / 14%), 0 3px 2px
  hsl(0deg 0% 0% / 4%), 0 7px 7px
  hsl(0deg 0% 0% / 3%);
```

Console | Performance monitor

CPU usage	0.1%
JS heap size	144 MB
DOM Nodes	102,628
JS event listeners	
Documents	
Document Frames	
Layouts / sec	
Style recalcs / sec	

13:13:10 13:13:20 13:13:30 13:13:40 13:13:50 13:14:00 13:14:10

CPU usage 100% 50%

JS heap size 200 MB 100 MB

DOM Nodes 400,000 200,000





Рубрика “Эксперимент”

Воркеры + фреймворк =
«Мощь, которая и не снилась моему отцу»





Воркеры + фреймворк= «Мощь, которая и не снилась моему отцу»

Backend

Framework

Shared VDom Worker

Shared App Worker

Your App(s)

Shared Data Worker

Main Thread (Browser Window) DOM OPs

Main Thread (Browser Window) DOM OPs

Main Thread (Browser Window) DOM OPs



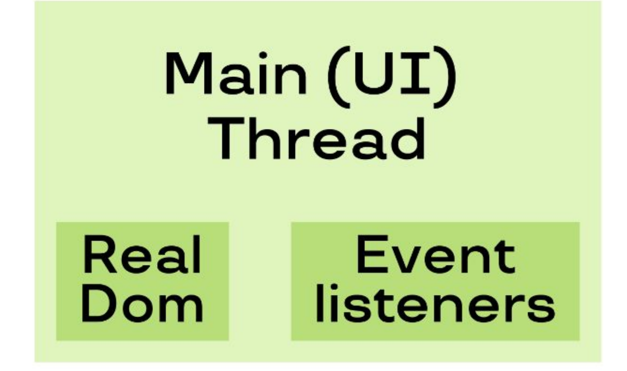
Архитектура фронтенд-приложений будущего

Main (UI) Thread
Real Dom
Event listeners
Event triggered
Render
Dedicated worker
VDom
Event hanlers
call Business Logic actions
App State Changed
Main (UI) Thread
Real Dom
Event listeners
Event triggered
Render
Dedicated worker
VDom
Event hanlers
call BL actions
App State Changed
Main (UI) Thread
Real Dom
Event listeners
Event triggered
Render
Dedicated worker
VDom
Event hanlers
call BL actions
App State Changed
Main (UI) Thread
Real Dom
Event listeners
Event triggered
Render
Dedicated worker
VDom
Event hanlers
call BL actions
App State Changed
Backend
DBMS
WS
fetch
Shared worker
WS
Business logic
AppState
fetch
Service worker
Interceptors
Configs
Auth
API
Adapter
update
Indexed DB
get/set Data



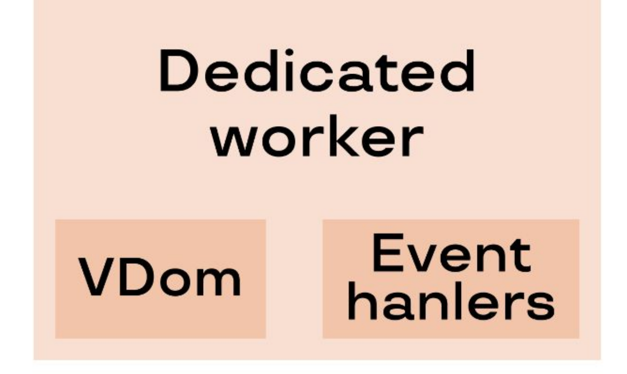


Основной поток (вкладка браузера)





Поток виртуального рендеринга





Поток бизнес-логики







Поток – сетевой адаптер







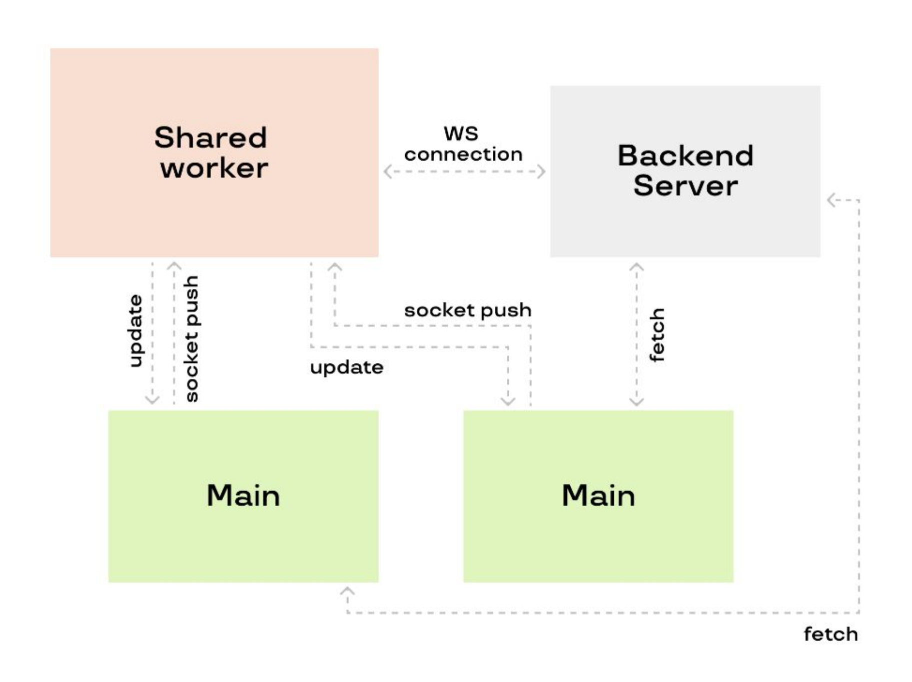
Архитектура фронтенд-приложений будущего





А может, мы уже в будущем?

WebSocket +
SharedWorker



А может, мы уже в будущем?

Stackblitz – WebContainers

- Faster than your local environment
- Node.js debugging in-browser
- Zero latency. Works offline
- Secure by default



Parallel Store – Demo



listen.worker.ts

```typescript
context.portList.add(port);

port.addEventListener(
  'message',
  (event: MessageEvent<MessageToWorkerData>) => {
    const { op, storeName = 'DEFAULT', key } = event.data;

    if (!(storeName in context.stores)) {⋯
    }

    const currentStore = context.stores[storeName];

    if (op === 'GET') {⋯
    } else if (op === 'SET') {⋯
    } else if (op === 'RUN_ACTION') {
      const { args } = event.data;

      if (key in currentStore.actions) {
        currentStore.actions[key](currentStore.data, ...args);
      }
    } else if (op === 'CREATE_ACTION') {
      if (!(key in currentStore.actions)) {
        const { source } = event.data;

        if (source.startsWith('function')) {
          currentStore.actions[key] = Function(
            `"use strict";return(${source})`
          )();
        } else {
          currentStore.actions[key] = Function(
            `"use strict";return( function ${source})`
          )();
        }
      }
    }
  }
```

defineStore.ts

```ts
import { inject, Ref, customRef } from 'vue';
import { ParallelStoreGlobal } from './@types/ParallelStoreGlobal';

import { parallelStoreInjectionToken } from './injectionToken';

const buildRef = <T>(
  parallelStore: ParallelStoreGlobal,
  storeName: string,
  key: string,
  initialValue: T
): Ref<T> => {
  let refTrigger: () => void;
  let value: T = initialValue;
  const ref: Ref<T> = customRef((track, trigger) => {
    refTrigger = trigger;
    return {
      get() {
        track();
        return value;
      },
      set(newValue) {
        value = newValue;
        parallelStore.worker.postMessage({ op: 'SET', storeName, key, value });
        trigger();
      },
    };
  });
  (ref as any).sync = (newValue: T) => {
    value = newValue;
    refTrigger();
  };
  parallelStore.worker.postMessage({ op: 'GET', storeName, key, value });

  return ref;
};
```

```
import { Ref } from 'vue';
import { defineStore } from '../plugins/ParallelStore';

interface CounterStoreState {
  counter: Ref<number>;
}

export const useCounterStore = defineStore(
  'CounterStore',
  () => ({ counter: 0 }),
  {
    incrementAction(state: CounterStoreState, value = 1) {
      state.counter.value += value;
    },
    incrementLaterAction(state: CounterStoreState, value = 1) {
      setTimeout(() => {
        state.counter.value += value;
      }, 1000);
    },
  }
);
```

FirstPage.vue

```
<script setup>
import { ref } from 'vue';
import { useWorkerRef } from '../plugins/ParallelStore';
import { useCounterStore } from '../store/counter';

defineProps({
  msg: String,
});

const count = useWorkerRef('count', 0);

const counterStore = useCounterStore();
</script>

<template>
  <h1>{{ msg }}</h1>

  <div class="card">
    <span>counterStore.counter is {{ counterStore.counter }}</span>
    <br />
    <button type="button" @click="counterStore.counter++">
      Update ref by increment
    </button>
    <button type="button" @click="counterStore.incrementAction()">
      update by action
    </button>
    <button type="button" @click="counterStore.incrementLaterAction(11)">
      update by async action (after 1 sec)
    </button>
```

Заключение

Feature	Usage % of	Global
Service Workers - CR	all users	96.59% + 0.08% = 96.67%
Shared Web Workers - LS	tracked desktop	89.29%
Shared Web Workers - LS	tracked mobile	3.94%
Web Workers - LS	all users	98.41%



Полезные ссылки

- Ссылка на Demo – Stackblitz, Github
- Выступление *Surma* на Chrome Dev Summit 2019 –
- The main thread is overworked & underpaid
- *Surma* для Smashing Magazine – The State Of Web Workers In 2021
- Comlink – Github
- Neo.mjs – Github, Blog & examples
- *Ayush Gupta* для dev.to – Scaling WebSocket Connections using Shared Workers
- Stackblitz – Introducing WebContainers: Run Node.js natively in your browser





Спасибо за внимание!



Игорь Костяков

Технический директор
IBS Dunice

@ikostyakov

i.kostyakov@dunice.net



